Community Focusing: Yet Another Query-Dependent Community Detection

Zhuo Wang,[1,3] **Weiping Wang,**[1*] **Chaokun Wang,**[2*] **Xiaoyan Gu,**[1] **Bo Li,**[1] **Dan Meng**[1]
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Software, Tsinghua University, Beijing, China
[3]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{wangzhuo, wangweiping}@iie.ac.cn, chaokun@tsinghua.edu.cn, {guxiaoyan, libo, mengdan}@iie.ac.cn

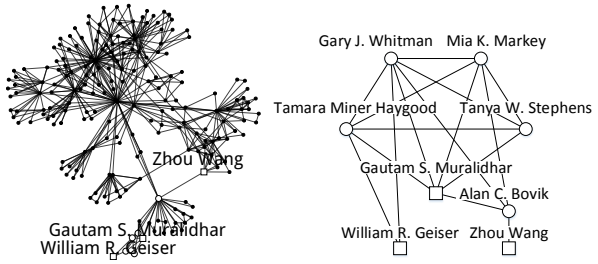
Abstract

As a major kind of query-dependent community detection, community search finds a densely connected subgraph containing a set of query nodes. As density is the major consideration of community search, most methods of community search often find a dense subgraph with many vertices far from the query nodes, which are not very related to the query nodes. Motivated by this, a new problem called community focusing (CF) is studied. It finds a community where the members are close and densely connected to the query nodes. A distance-sensitive dense subgraph structure called β-attention-core is proposed to remove the vertices loosely connected to or far from the query nodes, and a combinational density is designed to guarantee the density of a subgraph. Then CF is formalized as finding a subgraph with the largest combinational density among the β-attention-core subgraphs containing the query nodes with the largest β. Thereafter, effective methods are devised for CF. Furthermore, a speed-up strategy is developed to make the methods scalable to large networks. Extensive experimental results on real and synthetic networks demonstrate the performance of our methods.

Introduction

As a major kind of query-dependent community detection (Fang et al. 2016), *community search* finds a densely connected subgraph containing a set of given query nodes (Sozio and Gionis 2010). The problem has attracted great interest due to its wide applications in biological networks (Wu et al. 2015), tagging systems (Sozio and Gionis 2010), and social networks (Cui et al. 2014).

Unfortunately, as density is the major consideration of community search, most methods of community search often find a dense subgraph where many vertices are far from the query nodes and the query nodes stage at the periphery. As a result, the subgraph is not very related to the query nodes, and it is hard to interpret the cause of the subgraph. As a case study, we build a DBLP co-author network in which each edge between two authors indicates they have co-authored no less than three times (Huang et al. 2015; Wu et al. 2015). The network contains 242K vertices and 530K edges. Three scholars (Zhou Wang, Gautam S. Muralidhar and William R. Geiser) are selected from the network as the query nodes. They have the experience studying or working at the University of Texas, and articles published in J. Digital Imaging. Given the query nodes, the subgraph found by LCTC (a state-of-the-art method of community search) is shown in Fig.1(a). It is not very meaningful to the query nodes as: (1) The query nodes are marginalized in the subgraph. (2) The members in the subgraph come from various backgrounds. (3) Many members (60%) in the subgraph publish articles in NeuroComputing whereas none of the query nodes have articles published in the journal.



(a) LCTC (b) FLCF

Figure 1: Case study 1. In the subgraphs, the circles are the discovered members and the rectangles are the query nodes. As too many members exist in (a), only the query nodes are tagged with names. The white circles in (a) are the vertices that are also in (b).

Intuitively, a community meaningful to the query nodes should satisfy the following conditions: (1) The members of the community should be close to the query nodes. In general, the further a vertex is away from the query nodes, the less relevant the vertex is to the query nodes. (2) The members of the community should be densely connected to the query nodes, which requires strong associations to the query nodes. Therefore, a new problem called community focusing is proposed in this paper. Given a set of query nodes, we find a community where the members are close and densely connected to the query nodes. Different from community search, distance to the query nodes and density of a subgraph are equally important in our problem.

Our first contribution is proposing and formalizing the community focusing problem (CF). A distance-sensitive

*Corresponding authors: Weiping Wang and Chaokun Wang

dense subgraph structure called β-attention-core is devised to discover the vertices close and densely connected to the query nodes. As the structure cannot guarantee the density of a subgraph when β is small, combinational density is designed to enhance the density of a subgraph. Then the problem is formalized as finding a subgraph with large combinational density among the β-attention-core subgraphs containing the query nodes with the largest β values.

Our second contribution is devising effective methods for CF. Methods optimizing β and the combinational density of a β-attention-core subgraph are developed. Then a local method called LCF is proposed. The method builds a Steiner tree as a sketch containing the query nodes. Then it iteratively explores the neighborhood of the Steiner tree and uses the optimizing methods as subroutines to approach the optimal solution of CF.

Our third contribution is accelerating LCF. The demand for the connectivity of the query nodes makes the methods of community search inefficient. The problem is inherited in community focusing. Methods (Barbieri et al. 2015; Huang et al. 2015; Wu et al. 2015) for the community search build a Steiner tree to connect the query nodes. However, the usually adopted algorithm (Mehlhorn 1988) needs to visit the whole graph, which is time-consuming in large graphs. Considering the characteristic that the vertices in a community are not far from each other, a local method with provable guarantees is proposed to build the Steiner tree for LCF. As the proposed method need not visit the whole graph, LCF with the speed-up method is scalable to the large networks.

Our fourth contribution is experimenting with various methods on the real networks and synthetic networks. The results show that our methods are effective to discover the queried communities on the networks with ground-truth communities. Besides, LCF with the speed-up method is about two magnitudes faster than the state-of-the-arts of community search on average.

Related Work

Community search (Sozio and Gionis 2010) is a major kind of query-dependent community detection (Fortunato and Santo 2009; Wang et al. 2015; Xin et al. 2017). Studies (Cui et al. 2014; Huang et al. 2014; Cui et al. 2013; Akbas and Zhao 2017) find communities with a single query vertex. Given a set of query nodes, studies (Sozio and Gionis 2010; Barbieri et al. 2015; Huang et al. 2015; Wu et al. 2015; Yuan et al. 2018) discover a densely connected subgraph containing the nodes. Some variants of community search, such as attributed (Huang and Lakshmanan 2017; Fang et al. 2016; Shang et al. 2017), influential (Li et al. 2015), spatial (Fang et al. 2017) and skyline (Li et al. 2018) community search, have also been studied recently.

In community search, studies (Sozio and Gionis 2010; Barbieri et al. 2015; Huang et al. 2015; Wu et al. 2015; Yuan et al. 2018) handling multiple query nodes are similar to ours. MDC (Sozio and Gionis 2010) discovers a connected k-core subgraph containing the query nodes with the largest k under a size constraint. GrCon (Barbieri et al. 2015) outputs the subgraph with the minimum vertices among the connected k-core subgraphs with the largest k. QDC (Wu et al. 2015) finds a connected query-biased weighted subgraph with the largest weighted density. LCTC (Huang et al. 2015) finds a subgraph with the smallest diameter among the connected k-truss subgraphs with the largest k. DCPC (Yuan et al. 2018) finds a maximal connected k-cliques with the largest k. Compared to these works, our work has the following features: (1) Community focusing is a new problem. Different from community search, distance is considered in our problem, which avoids the marginalization of the query nodes. (2) The β-attention-core is a distance-sensitive dense subgraph structure, which discovers the vertices close and densely connected to the query nodes. Although MDC, QDC and LCTC constrain the size of a community, they have limitations. MDC and LCTC maximize the k of a k-core/truss subgraph in priority, which may miss the vertices close to the query nodes, and make the vertices far from the query nodes hard to remove due to the k-core/truss constraint. In QDC, as the weights of the query nodes are large, the subgraph with the largest weighted density contains very few non-query vertices. (3) Combinational density ensures the density of a subgraph. When the largest k of a connected k-core/truss/clique subgraph is small, the subgraph may be loosely connected. The combinational density solves the problem. (4) LCF with the speed-up strategy (FLCF) is an efficient method without indices. Hence FLCF is scalable to large networks without preprocessing.

Using the Steiner tree as a sketch for local exploration is widely adopted in community search (Sozio and Gionis 2010; Barbieri et al. 2015; Huang et al. 2015; Wu et al. 2015; Huang and Lakshmanan 2017). 2-approximate methods (Kou, Markowsky, and Berman 1981; Mehlhorn 1988) are widely applied to the NP-hard problem. However, the methods still visit the whole graph, which is time-consuming. Considering that query nodes in the same community are close to each other, we develop a speed-up strategy to build the sketch. Compared to the index-based (He et al. 2007; Gubichev and Neumann 2012; Li et al. 2008) and breadth-first (Kacholia et al. 2005; Kasneci et al. 2009; Bhalotia et al. 2002) heuristics, our strategy admits 2-approximation to the Steiner tree problem by visiting tiny vertices to connect the query nodes without indices.

Several studies find interesting subgraphs with a set of query nodes. Studies (Tong and Faloutsos 2006; Ruchansky et al. 2015; Faloutsos, Mccurley, and Tomkins 2004) find a subgraph connecting the query nodes. Studies (Gionis, Mathioudakis, and Ukkonen 2015; Ruchansky et al. 2017) discover the main components of the query nodes. Different from these works, our work not only provides good connections among the query nodes but also discovers a community meaningful to the query nodes.

Problem Statement

The community focusing problem is formulated on a connected graph $G(V, E)$ where V is the set of vertices, and E is the set of edges. Table 1 lists the commonly used notations in this paper.

Table 1: Main Symbols

Symbol	Definition
$G(V, E)$	A connected and undirected graph
$Q; q$	A set of query nodes Q; a query node q
$N_G(v)$	The neighbors of v in a graph $G(V, E)$
$deg_G(v)$	The degree of v in a graph $G(V, E)$
$dist_G(u, v)$	The length of the shortest path from u to v in a graph $G(V, E)$
$att_{G_s,Q}(v)$	The attention of a vertex in a subgraph G_s with Q
$ma(G_s, Q)$	The minimum attention of the vertices in a subgraph G_s with Q, i.e., $ma(G_s, Q)$= $min\{att_{G_s,Q}(v)\|v \in V(G_s)\}$
$cd(G_s, \alpha)$	The combinational density of a subgraph G_s and a tuning factor α

Distance-sensitive dense subgraph structure

Definition 1 *(Negligible Vertex). Given a graph $G(V, E)$ and a set of query nodes Q, a vertex v is a negligible vertex of Q if $\exists u \in N_G(v)$ makes that $\forall q \in Q$, $dist_G(v, q) = dist_G(u, q) + 1$.*

Given a negligible vertex v, v has a neighbor u that for each query node $q \in Q$, u is on a shortest path between v and q. In Fig.2(a), c_1 is a negligible vertex as v_2 is on the shortest path between c_1 and q_1, and the shortest path between c_1 and q_2. Similarly, c_2, c_3, and c_4 are also negligible vertices.

In real networks, the relationship between v_2 and v_3, and the relationship between v_2 and c_2 in Fig.2(a) may be different. For instance, q_1, q_2, v_1, v_3 are v_2's families, and c_1, c_2, c_3, c_4 are v_2's workmates. The relationship between v_1 and c_2 is built via v_2.

In our problem, the negligible vertices are removed as they are not densely connected or close enough to the query nodes. For instance, in Fig.2(a), c_1 should communicate with the query nodes via v_2. Besides, v_2 is closer to any query node than c_1.

Given a graph $G(V, E)$ and a set of query nodes Q, the focusing level (fl) of a vertex v to Q is defined as follows:

$$fl_{G,Q}(v) = \frac{1}{\sum_{q \in Q} dist_G(v, q)} \tag{1}$$

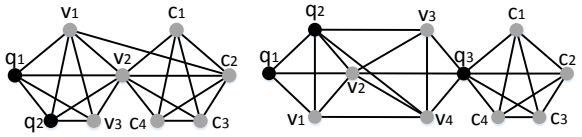
fl measures the closeness of a vertex to the query nodes. The closer a vertex is to the query nodes, the larger the fl of the vertex is. For example, in Fig.2(b), $fl_{G_2,Q}(v_1) = \frac{1}{1+1+2} = 0.25$ and $fl_{G_2,Q}(c_1) = \frac{1}{3+3+1} = 0.14$.

In order to discover the vertices close and densely connected to the query nodes, a distance-sensitive dense subgraph structure called β-attention-core is designed.

Given a graph $G(V, E)$ and a set of query nodes Q, the weight of an edge (u, v) in $G(V, E)$ is defined as follows:

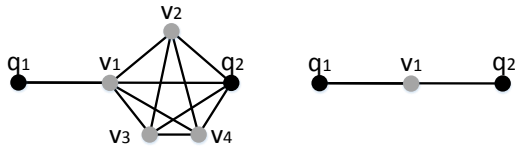
$$w_G(u, v) = \begin{cases} 0, & u \in F \vee v \in F; \\ 1, & u \notin F \wedge v \notin F; \end{cases} \tag{2}$$

F is the set of negligible vertices of Q in $G(V, E)$. For each edge adjacent to a negligible vertex, its weight is set to 0.



(a) G_1, $Q = \{q_1, q_2\}$ (b) G_2, $Q = \{q_1, q_2, q_3\}$

Figure 2: Examples to illustrate β-attention-core.



(a) G_3, $Q = \{q_1, q_2\}$ (b) G_4, $Q = \{q_1, q_2\}$

Figure 3: β-attention-core subgraphs with the largest $\beta = 0.5$. The average degree of G_3 is 3.66 and that of G_4 is 1.33. The internal density of G_3 is 0.73 and that of G_4 is 0.66. When $\alpha = 0.5$, the combinational density of G_3 is 0.82 and that of G_4 is 0.47.

The attention of a vertex v in a subgraph $G_s (G_s \subseteq G)$ is defined as:

$$att_{G_s,Q}(v) = fl_{G,Q}(v) \cdot \sum_{u \in N_{G_s}(v)} w_G(u, v) \tag{3}$$

In Fig.2(b), $fl_{G_2,Q}(q_3) = \frac{1}{2+2+0} = 0.25$. In the subgraph $G_{s1} = G_2$ in Fig.2(b), $att_{G_{s1},Q}(q_3) = 0.25 \cdot 3 = 0.75$ (As c_1, c_2, c_3 and c_4 are negligible vertices, the total weight of the edges adjacent to q_3 is 3).

Definition 2 *(β-attention-core). Given a graph $G(V, E)$, a set of query nodes Q, and a number β, a connected subgraph H is called a β-attention-core if (1) $H \subseteq G$, (2) $\forall v \in V(H), att_{H,Q}(v) \geq \beta$.*

With the vertices c_1, c_2, c_3 and c_4 in Fig.2(b) removed, the resulted subgraph G_s ($G_s \subseteq G_2$) is a 0.75-attention-core as the minimum attention of the vertices in $V(G_2)$ is 0.75 ($att_{G_2,Q}(q_3) = \frac{3}{2+2+0} = 0.75$).

β-attention-core can be used to prune the vertices loosely connected to or far from the query nodes. When $\beta > 0$, the negligible vertices are removed as the attention of a negligible vertex is 0. Besides, if a vertex is far from the query nodes or has small degree, its attention is small. Therefore, by maximizing the β of a β-attention-core, vertices close and densely connected to the query nodes can be discovered.

Theorem 1 *Given a graph $G(V, E)$ and a set of query nodes Q, a β-attention-core G_o containing Q with the largest β contains no negligible vertices.*

Combinational Density

Given a connected β-attention-core containing the query nodes with the largest $\beta = \beta_{max}$, its density cannot be guaranteed when β_{max} is small. For instance, both G_3 and G_4 in Fig.3 are β-attention-core subgraphs with the largest $\beta = 0.5$, however, G_4 is sparser than G_3.

The internal density (Yang and Leskovec 2015) of a subgraph $S(V, E)$ is $\frac{2\cdot|E(S)|}{|V(S)|\cdot(|V(S)|-1)}$, which measures the density of S. A clique is a subgraph maximizing the internal density. However, as cliques with large size are not common in real networks, the subgraph containing the query nodes with the largest internal density tends to be small in size. In order to find a dense subgraph with proper size, a tuning factor is provided to relax the internal density.

Given a subgraph $S(V, E)$ and a number α ($0 \leq \alpha \leq 1$), the combinational density (cd) of S is defined as follows:

$$cd(S, \alpha) = \frac{2 \cdot |E(S)|}{|V(S)| \cdot (|V(S)| - 1)^{\alpha}} \tag{4}$$

$cd(S, \alpha)$ can be treated as a combination of the average degree and internal density. When $\alpha = 0$, $cd(S, \alpha)$ is the average degree. When $\alpha = 1$, $cd(S, \alpha)$ is the internal density.

Problem Definition

The community focusing problem (CF) is formalized as:

Problem 1 ***(CF)****. Given a graph $G(V, E)$, a set of query nodes $Q \subseteq V(G)$, and a number α ($0 \leq \alpha \leq 1$), find a subgraph G_s of G, such that*

(1) G_s is a β-attention-core containing Q with the largest β;

(2) G_s has the largest $cd(G_s, \alpha)$ among the subgraphs satisfying condition (1).

The single-vertex-query is handled individually. Assume $Q = \{q\}$, for each neighbor v of q, $Q' = \{v, q\}$ is generated as the input of CF. As a result, the solution to the single-vertex-query is a set of subgraphs. Two reasons account for the above operation: (1) When $Q = \{q\}$, any vertex except q is a negligible vertex to be removed. The neighbors of q are used to make CF suitable for the single-vertex-query. (2) Given a query node q, it may belong to many communities (Cui et al. 2013). The neighbors of q help to discover communities with different semantics. For instance, if a neighbor v is $q's$ mother, a community consisting of q's families is discovered. If v is q's workmate, a discovered community consists of $q's$ workmates.

Method

Optimizing β of a β-attention-core

Optimizing β of a β-attention-core G_s is equal to maximizing the minimum attention of the vertices in G_s. Indeed, the attention of a vertex is a node-monotone non-increasing function (Sozio and Gionis 2010).

Theorem 2 *The attention of a vertex is node-monotonic non-increasing.*

Hence a β-attention-core G_s ($Q \subset V(G_s)$) with the largest β can be obtained by iteratively removing the vertex with the minimum attention. A method called $OPT\text{-}ma$ is designed to obtain a maximal β-attention-core containing the query nodes with the largest β:

Given a graph $G(V, E)$ and a set of query nodes Q, the details of $OPT\text{-}ma$ are the following:

1. Compute the attention of each vertex in $V(G)$.
2. $l = 0$, $G_l = G$. The vertex with the minimum attention is removed from G_l iteratively:
   2.1. Obtain a vertex u with the minimum attention from G_l, that is, $\forall v \in V(G_l)$, $att_{G_l,Q}(u) \leq att_{G_l,Q}(v)$.
   2.2. Obtain G_{l+1} by deleting u from G_l, $l = l + 1$.
   2.3. The attention of each u's neighbor is updated, i.e., $\forall v \in N_{G_{l-1}}(u), att_{G_l,Q}(v) = fl_{G,Q}(v) \cdot \sum_{v' \in N_{G_l}(v)} w_G(v, v')$.
   2.4. If Q is not connected in G_l or any query node has the minimum attention in G_l, return the subgraph $R = \arg\max_{G_t}\{ma(G_t, Q)|t = 0, 1, ..., l - 1\}$.

Theorem 3 *$OPT\text{-}ma$ runs in $O(|Q|m + nm' + mlogn)$ where $n = |V(G)|$, $m = |E(G)|$, and m' ($m' \ll m$) is the number of visited edges to check whether the query nodes are connected in a subgraph.*

Optimizing the combinational density of a β-attention-core

Given a β-attention-core G_s containing a given set of query nodes Q, Lagrange multipliers method (Bertsekas 2014) is used to optimize the combinational density of G_s while maintaining the property of β-attention-core.

Suppose G_s is represented by an adjacency matrix A_{ij} where $A_{ij} = 1$ if vertex v_i is adjacent to v_j and $A_{ij} = 0$ otherwise. Let $n = |V(G_s)|$ be the number of vertices in G_s, and $\vec{x} \in \{0, 1\}^n$ be a binary indicator vector representing a subset of $V(G_s)$. Given a subgraph G_o of G_s, $V(G_o)$ can be represented by $\vec{x}$ with the vertices in G_o set to 1s. Then the combinational density of G_o is:

$$cd(G_o, \alpha) = \frac{\vec{x}A\vec{x}^T}{\vec{x}^T\vec{x} \cdot (\vec{x}^T\vec{x} - 1)^{\alpha}} \tag{5}$$

where $\vec{x}^T$ is the transpose of $\vec{x}$. $\vec{x}^T\vec{x}$ is the number of vertices in G_o, and $\vec{x}A\vec{x}^T$ is twice the number of edges in G_o.

If G_o is a β-attention-core, $\forall v \in V(G_o)$, v's attention should be no less than β:

$$\vec{x}_v W A \vec{x}^T \geq \beta \tag{6}$$

where $\vec{x}_v \in \{0, 1\}^n$ is a one-hot encoded vector, i.e., only v in $\vec{x}_v$ is set to 1 and the other vertices are all set to 0s. W is an $n \times n$ diagonal matrix. If v is a negligible vertex, $W_{vv} = 0$. Otherwise, $W_{vv} = fl_{G,Q}(v)$.

If G_o contains all the query nodes Q:

$$\vec{x}_Q\vec{x}^T \geq |Q| \tag{7}$$

where $\vec{x}_Q \in \{0, 1\}^n$ is a vector with only the query nodes set to 1s.

Then the Lagrangian function which maximizes Equation 5 with the constraints Equation 6 and Equation 7 is:

$$\begin{aligned} L(\vec{x}, \vec{\lambda}, \mu) = &-\frac{\vec{x}A\vec{x}^T}{\vec{x}^T\vec{x} \cdot (\vec{x}^T\vec{x} - 1)^{\alpha}} + \\ &\sum_{v=1}^{n} \lambda_v(\beta - \vec{x}_v W A \vec{x}^T) + \mu(|Q| - \vec{x}_Q\vec{x}^T) \end{aligned} \tag{8}$$

where λ_v is the parameter of v in $\vec{\lambda}$.

Due to the inequality constraints (Equations 6 and 7), Karush-Kuhn-Trucker conditions should be satisfied:

$$\nabla_{\vec{x}} L(\vec{x}, \vec{\lambda}, \mu) = 0 \quad (9)$$

$$\mu(|Q| - \vec{x}_Q \vec{x}^T) = 0, \lambda_v(\beta - \vec{x}_v W A \vec{x}^T) = 0 \quad (10)$$

$$|Q| - \vec{x}_Q \vec{x}^T \leq 0, \beta - \vec{x}_v W A \vec{x}^T \leq 0 \quad (11)$$

$$\mu \geq 0, \lambda_v \geq 0 \quad (12)$$

v is any vertex that is set to 1 in $\vec{x}$.

In order to minimize Equation 8, we develop a method called $OPT\text{-}cd(G_s, Q, \alpha, \beta)$. The details are the following:

1. Initialize $\vec{x}$ with each element set to 1. $G_o = G_s$.
2. $\vec{x}$ is iteratively updated by the largest drop of Equation 8:
   2.1. Let X be the vertices that are set to 1s in $\vec{x}$. For each vertex $u \in X$, let G_u be the subgraph which removes u from G_o. Then $OPT\text{-}ma$ is used to find a β-attention-core R_u containing the query nodes from G_u.
   2.2. Denote $\vec{u}$ as a vector of length n with only the vertices in R_u set to 1s. Find the vector $\vec{u}_{max}$ which achieves the largest positive drop of Equation 8 among the vectors $\{\vec{u}|u \in X\}$. $\vec{x} = \vec{u}_{max}$. Remove the vertices which are set to 0s in $\vec{x}$ from G_o. As $\vec{x}$ and $\{\vec{u}|u \in X\}$ are satisfied with the constraints (Equations 6 and 7), the drop of Equation 8 is equal to $-\frac{\vec{x}A\vec{x}^T}{\vec{x}^T\vec{x}\cdot(\vec{x}^T\vec{x}-1)^\alpha} + \frac{\vec{u}A\vec{u}^T}{\vec{u}^T\vec{u}\cdot(\vec{u}^T\vec{u}-1)^\alpha}$.
   2.3. If there are no feasible solutions for $OPT\text{-}ma$ in Step 2.1 or positive drops in Step 2.2, G_o is returned.

Theorem 4 *$OPT\text{-}cd$ runs in $O(m_s n_s^2)$ where $n_s = |V(G_s)|$, $m_s = |E(G_s)|$.*

LCF

A straightforward method to solve the CF problem is applying $OPT\text{-}ma$ and $OPT\text{-}cd$ orderly. However, as $OPT\text{-}ma$ and $OPT\text{-}cd$ remove vertices from the input graph iteratively, the method is inefficient in large graphs. For efficiency, a local method called LCF is proposed.

Given a graph $G(V, E)$, a set of query nodes Q, the tuning factor α of the combinational density, and a size constraint η, LCF works as follows:

1. Build a Steiner tree T containing Q. T is used as a sketch for expansion. The 2-approximate method (Mehlhorn 1988) is adopted for this NP-hard problem.
2. Obtain a subgraph G_{can} induced from G with $V(T)$.
3. Enlarge $ma(G_{can}, Q)$ and $cd(G_{can}, \alpha)$ iteratively:
   3.1. Expand G_{can} with vertices having large attentions and ensure that $|V(G_{can})| \leq \eta$. In the neighborhood of G_{can}, vertices with attentions no less than $ma(G_{can}, Q)$ are added into G_{can}. Denote S as the set of vertices to add. If $|S \cup V(G_{can})| > \eta$, only $\eta - |V(G_{can}|$ vertices with larger attentions are selected. Given a vertex v, as $att_{G,Q}(v)$ is unknown, $att_{G[V(G_{can} \cup \{v\}],Q}(v)$ is used to approximate $att_{G,Q}(v)$. $G[V(G_{can} \cup \{v\}]$ is the subgraph induced by $V(G_{can}) \cup \{v\}$ from G.
   3.2. Enlarge $ma(G_{can}, Q)$ using $OPT\text{-}ma(G_{can}, Q)$.
   3.3. Use $OPT\text{-}cd(G_{can}, Q, \alpha, ma(G_{can}, Q))$ to enlarge the combinational density. Instead of computing the drop of $L(\vec{x}, \vec{\lambda}, \mu)$ for each vertex (Step 2.1 in $OPT\text{-}cd$), $\vec{x}$ is updated by removing the vertex with the minimum attention.
   3.4. If $ma(G_{can}, Q)$ and $cd(G_{can}, \alpha)$ are no longer increased, G_{can} is returned.

Theorem 5 *Denote t as the number of iterations (Step 3) in LCF and $G'(V', E')$ as the largest G_{can} during the loops. Then LCF runs in $O(mlogn + t|Q|m' + tm'n')$ where $m = |E(G)|$, $n = |V(G)|$, $m' = |E(G')|$, and $n' = |V(G')|$.*

LCF solves CF effectively due to the following reasons: (1) The Steiner tree T containing the query nodes Q can be a good sketch of a community. As T contains the minimum additional vertices to ensure the connectivity of Q, the additional vertices are probably close and densely connected to Q. Hence $V(T)$ are good seeds for expansion. (2) LCF approaches the solution of $OPT\text{-}ma$ by local expansion (Step 3.1). As the β-attention-core is a distance-sensitive dense subgraph structure, almost all the vertices in the β-attention-core with the largest β are around the query nodes. (3) Steps 3.1 and 3.3 increase the combinational density. The larger the degree of a vertex is in a β-attention-core, the larger attention the vertex has. In Step 3.1, vertices with large attentions are added and the constraint size limits a subgraph in a proper size. In Step 3.3, vertices with small attentions tend to be removed. Both the steps try to enlarge the number of edges and decrease the number of vertices in a subgraph, which increases the combinational density.

Speed-up Strategy

In LCF, a Steiner tree containing the query nodes is constructed using Mehlhorn's method (Mehlhorn 1988). As the method visits the whole graph, LCF is time-consuming in large graphs. In order to speed up building a Steiner tree, a strategy which only visits a part of the graph but with provable guarantees is developed.

Our strategy is in line with Mehlhorn's method (Mehlhorn 1988). Given a graph $G(V, E)$ and a set of query nodes Q, the method consists of the following steps:

A. Compute the partition of $V(G)$, i.e., $V(G) = \cup_{q \in Q} n(q)$. $n(q)$ denotes the set of vertices in $V(G)$, which are closer to a query node $q \in Q$ than the other query nodes. It is required that $\forall q_1, q_2 \in Q, q_1 \neq q_2, n(q_1) \cap n(q_2) = \emptyset$.
B. Create an auxiliary graph $G_1(V_1, E_1, D_1)$. $V_1 = Q$ is the set of query nodes. $E_1 = \{(q_s, q_t)|q_s, q_t \in Q \wedge \exists(u, v) \in E, u \in n(q_s), v \in n(q_t)\}$ and $D_1((q_s, q_t)) = min\{dist_G(q_s, u) + dist_G(u, v) + dist_G(v, q_t)|(u, v) \in E(G), u \in n(q_s), v \in n(q_t)\}$.
C. Find a minimum spanning tree T_1 of G_1 ($T_1 = MST(G_1)$). Build G_2 by replacing each edge in T_1 with its corresponding shortest path in G.
D. $T_2 = MST(G_2)$. Build a Steiner tree T_3 by deleting edges from T_2 to make the leaves are query nodes.

The method admits 2-approximation to the Steiner tree problem and runs in $O(|E(G)|log|V(G)|)$ if a binary heap is applied in Step A. Steps A and B are the most time-consuming steps in the method. In the steps, all the vertices and edges in $G(V, E)$ should be traversed. As a result, the time cost of the method is unaffordable in large networks.

Table 2: Network statistics(K=10^3 and M=10^6)

Network	Abbr.	$\|V\|$	$\|E\|$	Diameter
Amazon	AZ	335K	926K	44
DBLP	DP	317K	1M	21
Youtube	YT	1.1M	3M	20
LiveJournal	LJ	4M	35M	17
Orkut	OR	3.1M	117M	9

Our speed-up strategy is inspired by two observations: (1) Given a graph, the vertices in the same community are not far from each other in general. Hence we consider visiting a subset of $V(G)$ to build a Steiner tree. (2) As our problem is studied on an unweighted graph, the breadth-first search is used to find one shortest path between two vertices.

In our strategy, $s(v)$ denotes the query node closest to a vertex v in $G(V, E)$. Steps A and B are replaced as:

A'. Initialize an auxiliary graph $G'(V', E', D')$ as $V' = Q$, and $E' = D' = \emptyset$. $S = \emptyset$ is the set of visited vertices. $Queue$ is a queue for breadth-first search. For each query node $q \in Q$, $n(q) = \{q\}$, $S = S \cup \{q\}$, and q is pushed into $Queue$ with $dist_G(s(q), q) = 0$.

B'. If $Queue$ is empty or G' is connected, G' is returned. Otherwise, a vertex v is popped from $Queue$ with $dist(s(v), v)$. For each neighbor u of v,

B'.1. If $u \notin S$: (1) Insert u into S. (2) Insert u into $n(s(v))$. (3) u is pushed into $Queue$ with $dist_G(s(u), u) = dist_G(s(v), v) + 1$.

B'.2. If $u \in S, s(u) \neq s(v), (s(u), s(v)) \notin E(G')$, adjoin $s(u)$ and $s(v)$ with length equal to $dist_G(s(u), u)+1+dist_G(s(v), v)$ in G'.

Theorem 6 *For each edge $(q_s, q_t) \in E(G')$, $D'((q_s, q_t)) = D_1((q_s, q_t)) = dist_G(q_s, q_t)$.*

Theorem 7 *A minimum spanning tree of G' is a minimum spanning tree of G_1.*

Theorem 8 *The speed-up strategy runs in $O(|E'|log|V'| + |Q|^2)$ where E' is the set of visited edges, and V' is the set of visited vertices. Q is the set of query nodes.*

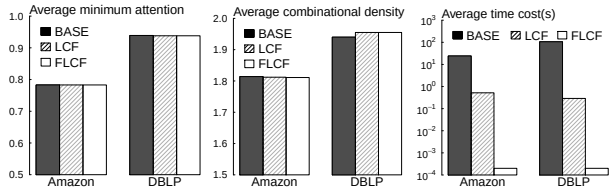
Compared to Mehlhorn's method, our strategy only visits a subset of the vertices in a graph. Once the query nodes are connected in the auxiliary graph, the strategy terminates. As the query nodes in the same community are not far from each other in a network, $|Q| \leq |V'| \ll |V(G)|$, $|E'| \ll |E(G)|$.

Experiments

Experimental Setup

Real-world networks are summarized in Table 2. They are publicly available from Stanford Network Analysis Project (*snap.stanford.edu*), and provide ground-truth communities.

Synthetic networks are generated using the LFR benchmark (Lancichinetti, Fortunato, and Radicchi 2008). Based on the default settings, n (number of vertices) and u (the proportion of neighbors around a vertex residing in other communities) are varied for evaluation. To model the communities of the real networks, 10% vertices belong to two or more communities, and the average degree is set to 10.



(a) Average minimum attention (b) Average combinational density (c) Average running time (s)

Figure 4: Comparison among BASE, LCF, and FLCF

Evaluation Criteria. $F_1 = \frac{2 \cdot |C \cap C_T|}{|C| + |C_T|}$ (Zhang, King, and Lyu 2015) measures the similarity between a discovered community C and the corresponding ground-truth community C_T.

Given a graph $G(V, E)$ and a subgraph $S(V, E)$, the conductance of S is $\frac{c_S}{2 \cdot m_S + c_S}$ where $m_S = |\{(u, v) \in E(S) : u \in V(S), v \in V(S)\}|$ and $c_S = |\{(u, v) \in E(G) : u \in V(S), v \notin V(S)\}|$. The lower the conductance is, the better separated S is from the rest of the network.

The geometric density $\frac{2 \cdot |E(S)|}{|V(S)| \cdot (|V(S)|-1)^{0.5}}$ of a subgraph $S(V, E)$ is the geometric mean of the average degree and the internal density. As is illustrated in Problem Statement, the subgraph with large internal density may be small in size. Thus a subgraph with large internal density may only contain the query nodes. In addition, a subgraph with large average degree may contain vertices not related to the query nodes (Wu et al. 2015). Hence the geometric density is used to measure the density of a subgraph while alleviating the drawbacks of the average degree and internal density.

Our methods. A straightforward method to solve CF is applying OPT-ma and OPT-cd sequentially. The method is called BASE for simplicity. LCF is a local method for CF. To accelerate LCF, a speed-up strategy is designed. FLCF (fast LCF) is called for LCF with the speed-up strategy.

Baselines. FLCF is compared with the state-of-the-arts (MDC (Sozio and Gionis 2010), GrCon (Barbieri et al. 2015), QDC (Wu et al. 2015), LCTC (Huang et al. 2015) and DCPC (Yuan et al. 2018)) of community search.

Environment. The methods are written in C++. The experiments are conducted on a Linux Server with 128GB main memory and Intel Xeon CPU E5-2630 (2.4GHz).

Evaluating the proposed methods

BASE, LCF, and FLCF are evaluated in terms of the minimum attention of a resulted subgraph, combinational density, and running time. As BASE is not scalable to large networks, the methods are compared on Amazon and DBLP. Using the drawn-by-drawn method (a simple random sampling method), 100 ground-truth communities are selected for each network. Then, $|Q|$ ($|Q| \in [1, 16]$) nodes are randomly selected as a query for each selected community.

Fig.4 shows the statistics when $\alpha = 0.5$ (the tuning factor of the combinational density) and $\eta = 200$ (the size constraint of LCF and FLCF). The methods are also tested with different α and η, and the tendencies of the results are similar to those in Fig.4.

Table 3: Average time costs(s) in real networks[a]

	A	B	C	D	E	F
AZ	**0.0003**	1.89	0.43	3.28	0.017	0.45
DP	**0.0003**	2.24	0.64	2.95	0.567	0.51
YT	**0.007**	8.1	2.82	10.8	1.39	1.27
LJ	**0.004**	54.6	18.3	112.8	-	16.3
OR	**0.018**	402.6	76.2	315.6	-	38.2

[a]A-F are FLCF, MDC, GrCon, QDC, DCPC, and LCTC orderly.

Table 4: Average $F_1(\%)$ in real networks[a]

	A	B	C	D	E	F
AZ	**92.8**	59.6	83.2	75.3	89.9	91.6
DP	**90.9**	44.5	82.0	75.1	67.6	86.2
YT	69.9	30.7	48.4	**75.6**	40.1	61.8
LJ	**81.5**	62.1	68.0	62.5	-	79.0
OR	**53.2**	41.4	42.8	40.2	-	39.4

[a]A-F are FLCF, MDC, GrCon, QDC, DCPC, and LCTC orderly.

Table 5: Average geometric density in real networks[a]

	A	B	C	D	E	F
AZ	**1.71**	1.21	1.64	1.38	1.62	1.69
DP	**1.99**	1.41	1.85	1.54	1.62	1.93
YT	**1.25**	0.94	1.17	1.23	0.72	1.1
LJ	**3.67**	3.38	3.23	2.31	-	3.61
OR	**3.6**	2.89	3.45	3.36	-	3.06

[a]A-F are FLCF, MDC, GrCon, QDC, DCPC, and LCTC orderly.

Table 6: Average conductance in real networks[a]

	A	B	C	D	E	F
AZ	**0.132**	0.195	0.208	0.416	0.137	0.15
DP	**0.408**	0.496	0.469	0.621	0.412	0.41
YT	0.831	0.905	0.881	0.844	0.869	**0.825**
LJ	**0.338**	0.419	0.491	0.675	-	0.359
OR	**0.718**	0.732	0.735	0.809	-	0.74

[a]A-F are FLCF, MDC, GrCon, QDC, DCPC, and LCTC orderly.

Fig.4(a) presents the average minimum attention. The methods achieve similar results, which shows that LCF and FLCF effectively enlarge the minimum attention of a subgraph by local expansion. Fig.4(b) shows the average combinational density. The methods produce similar average combinational density, which shows that the combinational density can be effectively enlarged by removing the vertex with the minimum attention iteratively. On DBLP, the combinational density of LCF and FLCF is slightly higher than that of BASE. In BASE, a vertex with large attention may be removed to increase the combinational density. As a result, the attention values of many vertices are decreased. Due to the constraint of β-attention-core, the combinational density reaches a small maximal value. In contrary, iteratively removing the vertex with the minimum attention in FLCF and LCF increases the combinational density continuously, which leads to a large maximal value. Fig.4(c) reports the average time costs. As both LCF and FLCF are local methods, they are faster than BASE. Compared to LCF, FLCF uses the speed-up strategy to build a Steiner tree which is equivalent to that of LCF. Hence FLCF is much faster than LCF and achieves high quality in the meanwhile.

Comparison with baselines

Evaluation in real networks. FLCF is compared with the baselines in the real networks. For each network, 1,000 communities are selected from the ground-truth communities using the drawn-by-drawn method. For each selected community, $|Q|$ ($|Q| \in [1, 16]$) nodes are randomly selected as a query. Then, the average running time, F_1, conductance, and geometric density are evaluated.

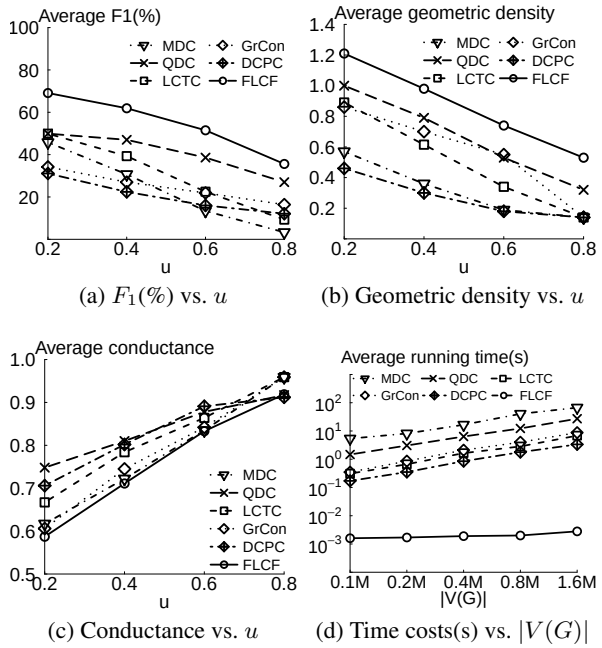
In FLCF, η is a size constraint of a resulted subgraph, and α is the tuning factor of the combinational density. Through empirical evaluation, η is set to 200. For each network, α is set to the number achieving the highest F_1 (α is set to 0.0, 0.8, 1.0, 0.4, and 0.2 for Amazon, DBLP, Youtube, LiveJournal, and Orkut respectively).

For a single-vertex-query, FLCF produces multiple communities. For fairness, the statistics are collected for $|Q| \in [2, 16]$. The number of such queries is about 850 for each network. DCPC is an index-based method. As the index size of LiveJournal and that of Orkut are too large to load into memory. We only report the DCPC results of Amazon, DBLP, and Youtube.

Table 3 lists the average time costs. Table 4, Table 5, and Table 6 present the average F_1, geometric density, and conductance of the methods respectively. Due to the speed-up strategy, FLCF is much faster than the other methods. Besides, FLCF outperforms the other methods in terms of the quality. For one thing, the β-attention-core discovers the vertices close and densely connected to the query nodes, which makes the resulted subgraph meaningful to the query nodes. Hence FLCF achieves higher F_1 compared to the other methods. For another, the combinational density guarantees the density of the resulted subgraph. Thus FLCF also outperforms the others in terms of geometric density and conductance. DCPC finds a maximal union of k-cliques containing the query nodes, which contains many irrelevant vertices and hence its F_1 is low. As the binary-tree-like index of DCPC is large and the bottom-up searching process ends at the node close to the root in many cases, DCPC is slower than FLCF. As the weights of the query nodes are large, the subgraph with the largest weighted density found by QDC contains very few non-query nodes. Thus QDC has large conductance. In Youtube, the ground-truth communities are sparse. As QDC finds a subgraph connecting the query nodes rather than a dense subgraph, QDC achieves high F_1 in Youtube.

Evaluation in synthetic networks. Based on the settings introduced in Experimental Setup, a set of LFR networks (Lancichinetti, Fortunato, and Radicchi 2008) are generated with different n and u. Query generation and evaluation are the same as the ones in the real networks (The number of queries with $|Q| > 1$ is about 950 for each network). α is set to 0 and η is set to 200 for FLCF.

A set of LFR networks are generated with $n = 100K$ and u varying from 0.2 to 0.8. Fig.5(a)-(c) show the average



(a) $F_1(\%)$ vs. u (b) Geometric density vs. u

(c) Conductance vs. u (d) Time costs(s) vs. $|V(G)|$

Figure 5: Evaluation on the synthetic networks

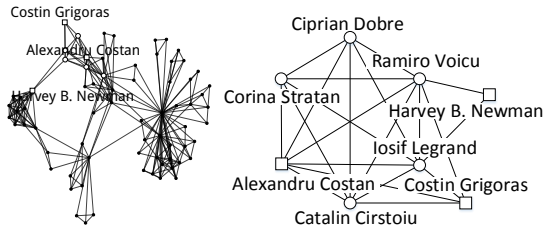
F_1, geometric density, and conductance respectively. Due to the same reason of the evaluation in real networks, FLCF discovers high-quality subgraphs. As u increases, the clearness of the community structure decreases. The ground-truth community becomes sparse and hard to detect. Hence the metrics worsen as u increases.

A set of LFR networks are generated with $u = 0.4$ and n varying from $0.1M$ to $1.6M$. Fig.5(d) reports the average time costs. As MDC, QDC, and LCTC visit the whole graph, their time costs become larger as n increases. When n is larger, GrCon deals with a larger subgraph, and the height of the binary-tree-like index in DCPC is larger. Hence GrCon and DCPC need more time to find a result when n is larger. As FLCF connects the query nodes locally, the time cost of FLCF increases with the increasing n slowly.

Case Studies

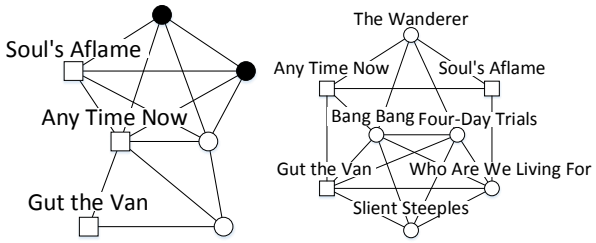
Case study 1. In Introduction, we give an example of LCTC on the co-author network of DBLP. The subgraph found by FLCF with the same query nodes is shown in Fig.1(b). The query nodes are close to the other members and centred. All the members have the experience studying or working at the University of Texas at Austin or the University of Texas MD Anderson Cancer Center. Besides, each member has articles published in J. Digital Imaging.

Case study 2. Three authors, Harvey B. Newman, Alexandru Costan and Costin Grigoras, are selected as a query. The authors have papers published in Computer Physics Communications, and are close to each other in the network. The LCTC-community is shown in Fig.6(a). Similar to Fig.1(a), the query nodes are marginalized and the members of the community come from various backgrounds. None of the



(a) LCTC (b) FLCF

Figure 6: Case study 2. The meaning of the colors and vertices is the same with that of Fig.1.



(a) LCTC (b) FLCF

Figure 7: Case study 3. The meaning of the vertices and colors is the same with that of Fig.1.

query nodes have papers published in the Journal of Mobile Information Systems which is popular in the community. Fig.6(b) is the community discovered by FLCF. They are the scholars coming from Polytechnic University Bucharest and California Institute of Technology. All of them have papers published in the same journal of the query nodes.

Case study 3. In the Amazon co-purchasing network with 548K nodes and 1.78M edges (Available from the SNAP project), three albums ("Any Time Now", "Gut the Van", and "Soul's Aflame") are input as the query nodes. The albums are tagged with "Rock Jam Bands". Fig.7(b) shows the community discovered by FLCF. All the albums are tagged with "Rock Jam Bands". Fig.7(a) is found by LCTC. One of the black circle is "Risen" which is tagged with "Alternative Rock". The other black circle is not related to music.

Conclusion

In this paper, community focusing is studied. Given a set of query nodes Q, it finds a subgraph where the vertices are close and densely connected to Q. We formalize the problem as finding the subgraph maximizing the combinational density among the β-attention-core subgraphs containing Q with the largest β. Then effective methods are developed. For scalability, a speed-up strategy is devised. Experiments demonstrate the performance of our methods. In the future, we will explore community focusing on attributed graphs.

Acknowledgments

This work was supported by the Strategic Priority Research Program of Chinese Academy of Sciences (No.

XDC02000000) and the National Natural Science Foundation of China (No. 61872207).

References

Akbas, E., and Zhao, P. 2017. Truss-based community search: a truss-equivalence based indexing approach. In *Proceedings of the VLDB Endowment*, volume 10(11), 1298–1308.

Barbieri, N.; Bonchi, F.; Galimberti, E.; and Gullo, F. 2015. Efficient and effective community search. *Data Mining and Knowledge Discovery* 29(5):1–28.

Bertsekas, D. P. 2014. Constrained optimization and lagrange multiplier methods. *Academic press*.

Bhalotia, G.; Hulgeri, A.; Nakhe, C.; Chakrabarti, S.; and Sudarshan, S. 2002. Keyword searching and browsing in databases using banks. *IEEE International Conference on Data Engineering* 431–440.

Cui, W.; Xiao, Y.; Wang, H.; Lu, Y.; and Wang, W. 2013. Online search of overlapping communities. In *ACM SIGMOD International Conference on Management of Data*, 277–288.

Cui, W.; Xiao, Y.; Wang, H.; and Wang, W. 2014. Local search of communities in large graphs. In *ACM SIGMOD International Conference on Management of Data*, 991–1002.

Faloutsos, C.; Mccurley, K. S.; and Tomkins, A. 2004. Fast discovery of connection subgraphs. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 118–127.

Fang, Y.; Cheng, R.; Luo, S.; and Hu, J. 2016. Effective community search for large attributed graphs. *Proceedings of the VLDB Endowment* 9(12):1233–1244.

Fang, Y.; Cheng, R.; Li, X.; Luo, S.; and Hu, J. 2017. Effective community search over large spatial graphs. *Proceedings of the VLDB Endowment* 10(6):709–720.

Fortunato, and Santo. 2009. Community detection in graphs. *Physics Reports* 486(3):75–174.

Gionis, A.; Mathioudakis, M.; and Ukkonen, A. 2015. Bump hunting in the dark: Local discrepancy maximization on graphs. In *IEEE International Conference on Data Engineering*, 1155–1166.

Gubichev, A., and Neumann, T. 2012. Fast approximation of steiner trees in large graphs. *ACM Internatinal Conference on Information and Knowledge Management* 1497–1501.

He, H.; Wang, H.; Yang, J.; and Yu, P. S. 2007. Blinks: ranked keyword searches on graphs. *ACM SIGMOD International Conference on Management of Data* 305–316.

Huang, X., and Lakshmanan, L. V. S. 2017. Attribute-driven community search. *Proceedings of the VLDB Endowment* 10(9):949–960.

Huang, X.; Cheng, H.; Qin, L.; Tian, W.; and Yu, J. X. 2014. Querying k-truss community in large and dynamic graphs. In *ACM SIGMOD International Conference on Management of Data*, 1311–1322.

Huang, X.; Lakshmanan, L. V. S.; Yu, J. X.; and Cheng, H. 2015. Approximate closest community search in networks. *Proceedings of the VLDB Endowment* 9(4):276–287.

Kacholia, V.; Pandit, S.; Chakrabarti, S.; Sudarshan, S.; Desai, R.; and Karambelkar, H. 2005. Bidirectional expansion for keyword search on graph databases. *Proceedings of the VLDB Endowment* 505–516.

Kasneci, G.; Ramanath, M.; Sozio, M.; Suchanek, F. M.; and Weikum, G. 2009. Star: Steiner-tree approximation in relationship graphs. *IEEE International Conference on Data Engineering* 868–879.

Kou, L.; Markowsky, G.; and Berman, L. 1981. A fast algorithm for steiner trees. *Acta Informatica* 15(2):141–145.

Lancichinetti, A.; Fortunato, S.; and Radicchi, F. 2008. Benchmark graphs for testing community detection algorithms. *Physical Review E* 78(4):046110.

Li, G.; Ooi, B. C.; Feng, J.; Wang, J.; and Zhou, L. 2008. Ease: an effective 3-in-1 keyword search method for unstructured, semi-structured and structured data. *ACM SIGMOD International Conference on Management of Data* 903–914.

Li, R. H.; Qin, L.; Yu, J. X.; and Mao, R. 2015. Influential community search in large networks. *Proceedings of the Vldb Endowment* 8(5):509–520.

Li, R. H.; Qin, L.; Ye, F.; Yu, J. X.; Xiao, X.; Xiao, N.; and Zheng, Z. 2018. Skyline community search in multi-valued networks. In *ACM SIGMOD International Conference on Management of Data*, 457–472.

Mehlhorn, K. 1988. A faster approximation algorithm for the steiner problem in graphs. *Information Processing Letters* 27(3):125–128.

Ruchansky, N.; Bonchi, F.; Garciasoriano, D.; Gullo, F.; and Kourtellis, N. 2015. The minimum wiener connector problem. *ACM SIGMOD International Conference on Management of Data* 1587–1602.

Ruchansky, N.; Bonchi, F.; Garciasoriano, D.; Gullo, F.; and Kourtellis, N. 2017. To be connected, or not to be connected: That is the minimum inefficiency subgraph problem. *ACM Internatinal Conference on Information and Knowledge Management* 879–888.

Shang, J.; Wang, C.; Wang, C.; Guo, G.; and Qian, J. 2017. An attribute-based community search method with graph refining. *The Journal of Supercomputing* 1–28.

Sozio, M., and Gionis, A. 2010. The community-search problem and how to plan a successful cocktail party. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 939–948.

Tong, H., and Faloutsos, C. 2006. Center-piece subgraphs: problem definition and fast solutions. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 404–413.

Wang, M.; Wang, C.; Yu, J. X.; and Zhang, J. 2015. Community detection in social networks: an in-depth benchmarking study with a procedure-oriented framework. *Proceedings of the VLDB Endowment* 8(10):998–1009.

Wu, Y.; Jin, R.; Li, J.; and Zhang, X. 2015. Robust local community detection: on free rider effect and its elimination. *Proceedings of the Vldb Endowment* 8(7):798–809.

Xin, X.; Wang, C.; Ying, X.; and Wang, B. 2017. Deep community detection in topologically incomplete networks. *Physica A Statistical Mechanics and Its Applications* 469:342–352.

Yang, J., and Leskovec, J. 2015. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems* 42(1):181–213.

Yuan, L.; Qin, L.; Zhang, W.; Chang, L.; and Yang, J. 2018. Index-based densest clique percolation community search in networks. *IEEE Transactions on Knowledge and Data Engineering* 30(5):922–935.

Zhang, H.; King, I.; and Lyu, M. R. 2015. Incorporating implicit link preference into overlapping community detection. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 396–402.